



USPTO

[Subscribe \(Full Service\)](#) [Register \(Limited Ser](#)

Search: ☒ The ACM Digital Library ☐ The

[+store +restore +instructions +register +determir](#)

[Feedback](#) [Report a problem](#)

Published since January 1985 and Published before September 2003

Terms used

store restore instructions register determine execution order before prior

Sort results
by

[Save results to a Binder](#)

Try an [Advanc](#)

[Search Tips](#)

Try this search

Display
results

☐ Open results in a new
window

Results 1 - 20 of 200

Result page: [1](#) [2](#) [3](#) [4](#) [5](#) [6](#) [7](#) [8](#) [9](#) [10](#) [ne](#)

Best 200 shown

Ra

1 [The priority-based coloring approach to register allocation](#)

Fred C. Chow, John L. Hennessy

October 1990 **ACM Transactions on Programming Languages and Sys**
Volume 12 Issue 4

Publisher: ACM Press

Full text available: [pdf\(2.97](#)
[MB\)](#)

Additional Information: [full citation](#), [abst](#)
[citing](#), [index ter](#)

Global register allocation plays a major role in determining the efficacy of a compiler. Graph coloring has been used as the central paradigm for register allocation in modern compilers. A straightforward coloring approach can suffer from several shortcomings. These shortcomings are addressed in this paper by coloring registers in priority ordering. A natural method for dealing with the spilling problem emerges. The detailed algorithms for a priority-based coloring approach are presented.

2 [Parallel execution of prolog programs: a survey](#)

Gopal Gupta, Enrico Pontelli, Khayri A.M. Ali, Mats Carlsson, Manuel V.

July 2001 **ACM Transactions on Programming Languages and System**
Volume 23 Issue 4

Publisher: ACM Press

Full text available: [pdf\(1.95](#)

Additional Information: [full citation](#), [abst](#)

MB)


citings, index ter

Since the early days of logic programming, researchers in the field realized exploitation of parallelism present in the execution of logic programs. The nature, the presence of nondeterminism, and their referential transparency characteristics, make logic programs interesting candidates for obtaining parallel execution. At the same time, the fact that the typical applications programming frequently involve irregular computation ...

Keywords: Automatic parallelization, constraint programming, logic programming, parallelism, prolog

3 Compiler transformations for high-performance computing

◆ David F. Bacon, Susan L. Graham, Oliver J. Sharp
December 1994 **ACM Computing Surveys (CSUR)**, Volume 26 Issue 4
Publisher: ACM Press

Full text available:  [pdf\(6.32 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


In the last three decades a large number of compiler transformations for logic programs have been implemented. Most optimizations for uniprocessors are based on scalar quantities and data-flow techniques. In contrast, optimizations for superscalar, vector, and parallel processors maximize parallelism and use transformations that rely on tracking the properties of ...

Keywords: compilation, dependence analysis, locality, multiprocessors, parallelism, superscalar processors, vectorization

4 An abstract machine for tabled execution of fixed-order stratified logic programs

◆ Konstantinos Sagonas, Terrance Swift
May 1998 **ACM Transactions on Programming Languages and Systems**
Volume 20 Issue 3

Publisher: ACM Press

Full text available:  [pdf\(602.38 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

SLG resolution uses tabling to evaluate nonfloundering normal logic programs with the well-founded semantics. The SLG-WAM, which forms the engine of the implementation, can compute in-memory recursive queries an order of magnitude faster than deductive databases. At the same time, the SLG-WAM tightly integrates tabling with SLG code, and executes Prolog code with minimal overhead compared to a pure SLG implementation. As a result, the SLG-WAM brings to logic programming ...


Keywords: SLG, WAM, memoing, prolog, stratification theories, tabling


5 Register integration: a simple and efficient implementation of squash reuse


◆ Amir Roth, Gurindar S. Sohi

December 2000 **Proceedings of the 33rd annual ACM/IEEE international conference on Computer-Aided Design (ICCAD)**
Microarchitecture

Publisher: ACM Press

Full text available:  [pdf\(154.98](#)

[KB\)](#)  [ps](#)
(573.81 KB)

 [Publisher](#)
[Site](#)


Additional Information: [full citation](#), [reference](#), [index terms](#)

6 Sentinel scheduling: a model for compiler-controlled speculative execution

◆ Scott A. Mahlke, William Y. Chen, Roger A. Bringmann, Richard E. Hank Hwu, B. Ramakrishna Rau, Michael S. Schlansker

November 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11, Number 4

Publisher: ACM Press

Full text available:  [pdf\(2.26](#)
[MB\)](#)

Additional Information: [full citation](#), [abstract](#), [citations](#), [index terms](#)

Speculative execution is an important source of parallelism for VLIW architectures. A serious challenge with compiler-controlled speculative execution is to efficiently handle exceptions for speculative instructions. In this article, we introduce features and compile-time scheduling support collectively referred to as Sentinel scheduling. Sentinel scheduling provides an effective framework for compiler-controlled speculative execution ...

Keywords: VLIW processor, exception detection, exception recovery, in-order execution


scheduling, instruction-level parallelism, speculative execution, supersca

7 Register renaming and dynamic speculation: an alternative approach

Mayan Moudgill, Keshav Pingali, Stamatis Vassiliadis


December 1993 **Proceedings of the 26th annual international symposium on Microarchitecture**

Publisher: IEEE Computer Society Press

Full text available:  [pdf\(1.49 MB\)](#)


Additional Information: [full citation](#), [reference](#)

8 Implementation of precise interrupts in pipelined processors

 James E. Smith, Andrew R. Pleszkun

June 1985 **ACM SIGARCH Computer Architecture News , Proceeding annual international symposium on Computer architecture**
13 Issue 3

Publisher: IEEE Computer Society Press, ACM Press

Full text available:  [pdf\(893.17 KB\)](#)


Additional Information: [full citation](#), [citation](#)

9 Experience with a software-defined machine architecture

 David W. Wall

May 1992 **ACM Transactions on Programming Languages and Systems**
Volume 14 Issue 3

Publisher: ACM Press

Full text available:  [pdf\(2.86 MB\)](#)

Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)

We have built a system in which the compiler back end and the linker w present an abstract machine at a considerably higher level than the actual intermediate language translated by the back end is the target language of compilers and is also the only assembly language generally available. The intermodule register allocation, which would be harder if some of the code had come from a traditional assembler, out of sight of ...


Keywords: RISC, graph coloring, intermediate language, interprocedural pipeline scheduling, profiling, register allocation, register windows

10 The MC88110 implementation of precise exceptions in a superscalar archi

◆ Nasr Ullah, Matt Holle

March 1993 **ACM SIGARCH Computer Architecture News**, Volume 2

Publisher: ACM Press

Full text available:  [pdf\(1.06 MB\)](#) Additional Information: [full citation](#), [abstract terms](#)


This paper describes the precise exception model of the MC88110 symmetric microprocessor. The MC88110 is a superscalar, pipelined processor that executes multiple execution units and allows out-of order execution of instructions. The MC88110 fully precise exceptions and presents the architecturally correct state to a handling routine in a manner that minimizes exception response latency. Exception latency timings in the MC88110 are described, and several ...

11 Exploiting instruction level parallelism in processors by caching scheduled

◆ Ravi Nair, Martin E. Hopkins

May 1997 **ACM SIGARCH Computer Architecture News**, **Proceeding annual international symposium on Computer architecture**
25 Issue 2

Publisher: ACM Press

Full text available:  [pdf\(2.01 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [citations](#), [index terms](#)


Modern processors employ a large amount of hardware to dynamically dispatch single-threaded programs and maintain the sequential semantics implied by the program. The complexity of some of this hardware diminishes the gains due to parallelism. A longer clock period or increased pipeline latency of the machine. In this paper, we present a processor implementation which dynamically schedules groups of instructions for execution on a fast simple engine and caches them for future use ...

12 Compiler techniques for code compaction

◆ Saumya K. Debray, William Evans, Robert Muth, Bjorn De Sutter

March 2000 **ACM Transactions on Programming Languages and Systems**
Volume 22 Issue 2


Publisher: ACM Press

Full text available:  [pdf\(409.20 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)


In recent years there has been an increasing trend toward the incorporation of code compaction into a variety of devices where the amount of memory available is limited. It is highly desirable to try to reduce the size of applications where possible. This article describes the use of compiler techniques to accomplish code compaction to yield smaller code. The main contribution of this article is to show that careful, aggressive, interprocedural optimization, together with procedural abstraction ...

Keywords: code compaction, code compression, code size reduction

13 VLIW compilation techniques in a superscalar environment

-  Kemal Ebcioglu, Randy D. Groves, Ki-Chang Kim, Gabriel M. Silberman, June 1994 **ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN Conference on Programming language design and implementation PLDI** Issue 6

Publisher: ACM Press

Full text available:  [pdf\(1.30 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)


We describe techniques for converting the intermediate code representation of a program, as generated by a modern compiler, to another representation that yields the same run-time results, but can run faster on a superscalar machine. The article describes novel parallelization techniques for Very Long Instruction Word (VLIW) architectures and places them together with independently executable operations that may be far from sequential. i.e., they may be sequential ...

Keywords: VLIW, compiler optimizations, global scheduling, profiling, software pipelining, superscalars

14 A novel renaming scheme to exploit value temporal locality through physical register allocation and unification

Stephen Jourdan, Ronny Ronen, Michael Bekerman, Bishara Shomar, Adi Shoshitaishvili, November 1998 **Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture**

Publisher: IEEE Computer Society Press

Full text available:  [pdf\(2.41 MB\)](#) Additional Information: [full citation](#), [reference terms](#)


Keywords: dependency redirection, physical register reuse, register and result reuse, value temporal locality

15 Optimization for a superscalar out-of-order machine

Anne M. Holler


December 1996 **Proceedings of the 29th annual ACM/IEEE international Microarchitecture**

Publisher: IEEE Computer Society

Full text available:  [pdf\(1.55 MB\)](#) Additional Information: [full citation](#), [abstract terms](#)


Compiler optimization plays a key role in unlocking the performance of innovative dynamically-scheduled machine which is the first implementation of the PA 2.0 member of the HP PA-RISC architecture family. This wide superscalar order machine provides significant execution bandwidth and automatical runtime; however, despite its ample hardware resources, many of the optimization transformations which proved effective for the PA-8000 served to au ...

16 Instruction-level reverse execution for debugging

 Tankut Akgul, Vincent J. Mooney

November 2002 **ACM SIGSOFT Software Engineering Notes , Proceedings of the ACM SIGPLAN-SIGSOFT workshop on Program analysis and tools and engineering PASTE '02, Volume 28 Issue 1**

Publisher: ACM Press


Full text available:  [pdf\(241.83 KB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

The ability to execute a program in reverse is advantageous for shortening the time a paper presents a reverse execution methodology at the assembly instruction level. This paper presents a reverse execution methodology at the assembly instruction level, memory and time overheads. The core idea of this approach is to generate code that is able to undo, in almost all cases, normal forward execution of an assembly program being debugged. The methodology has been implemented on a

in a custom made debugger. Compared to previous w ...

Keywords: debugging, reverse code generation, reverse execution



17 Software pipelining

- ◆ Vicki H. Allan, Reese B. Jones, Randall M. Lee, Stephen J. Allan
September 1995 **ACM Computing Surveys (CSUR)**, Volume 27 Issue 3
Publisher: ACM Press
Full text available:  [pdf\(4.72 MB\)](#) Additional Information: [full citation](#), [abstracts](#), [index terms](#)

Utilizing parallelism at the instruction level is an important way to improve performance. Because the time spent in loop execution dominates total execution time, compiler optimizations focus on decreasing the time to execute each iteration. Software pipelining is a technique that reforms the loop so that a faster execution rate is realized. It is executed in overlapped fashion to increase parallelism. Let $\{ABC\}^n$

Keywords: instruction level parallelism, loop reconstruction, optimization, software pipelining

18 Multithreading and value prediction: Speculative lock elision: enabling high multithreaded execution

- Ravi Rajwar, James R. Goodman
December 2001 **Proceedings of the 34th annual ACM/IEEE international symposium on Microarchitecture**
Publisher: IEEE Computer Society
Full text available:  [pdf\(1.37 MB\)](#)  Additional Information: [full citation](#), [abstracts](#), [index terms](#)
[Publisher](#) [Site](#)


Serialization of threads due to critical sections is a fundamental bottleneck in performance in multithreaded programs. Dynamically, such serialization is often unnecessary because these critical sections could have safely executed without locks. Current processors cannot fully exploit such parallelism because they lack mechanisms to dynamically detect such false inter-thread dependences. We present *Speculative Lock Elision (SLE)*, a novel micro-architectural technique ...

19 Avoidance and suppression of compensation code in a trace scheduling cor

◆ Stefan M. Freudenberger, Thomas R. Gross, P. Geoffrey Lowney

July 1994 **ACM Transactions on Programming Languages and System**
Volume 16 Issue 4

Publisher: ACM Press

Full text available:  [pdf\(3.58 MB\)](#) Additional Information: [full citation](#), [abst](#)
[citions](#), [index ter](#)

Trace scheduling is an optimization technique that selects a sequence of trace and schedules the operations from the trace together. If an operation crosses basic block boundaries, one or more compensation copies may be required. This article discusses the generation of compensation code in a trace compiler and presents techniques for limiting the amount of compensation code (restricting code motion so that no compensation ...


Keywords: SPEC89, instruction-level parallelism, performance evaluation

20 Dynamic dead-instruction detection and elimination

◆ J. Adam Butts, Guri Sohi

October 2002 **ACM SIGOPS Operating Systems Review**, **ACM SIGPLAN SIGARCH Computer Architecture News**, **Proceedings of the international conference on Architectural support for programming languages and operating systems ASPLOS-X**, Volume 36 Issue 10, 5




Publisher: ACM Press

Full text available:  [pdf\(1.50 MB\)](#) Additional Information: [full citation](#), [abst](#)
[citions](#)

We observe a non-negligible fraction--3 to 16% in our benchmarks--of *dead instructions*, dynamic instruction instances that generate unused results. These instructions arise from static instructions that also produce useful results. A compiler optimization (specifically instruction scheduling) creates a significant number of these *partially dead* static instructions. We show that most of the dynamic dead instructions arise from a small set of static instructions ...

The ACM Portal is published by the Association for Computing Machinery
ACM, Inc.

[Terms of Usage](#) [Privacy Policy](#) [Code of Ethics](#) [Contact](#)

Useful downloads:  [Adobe Acrobat](#)  [QuickTime](#)  [Windows Med
Player](#)

[Home](#) | [Login](#) | [Logout](#)


Welcome United States Patent and Trademark Office

Search Results

[BROWSE SEARCH](#) [IEEE GUIDE](#)

Results for "(((determine instruction order instrumenting store restore
<in>metadata)) <and> (pyr &g...
Your search matched 0 documents.
A maximum of 100 results are displayed, 25 to a page, sorted by Relevance
Descending order.

» Search Options

[View Session History](#)
[New Search](#)

Modify Search

(((determine instruction order instrumenting store re

☐ Check to search only within this results set

» Key

IEEE JNL IEEE
Journal or Magazine

IEEE JNL IEEE Journal
or Magazine

IEEE CNF IEEE
Conference Proceeding

IEEE CNF IEEE
Conference Proceeding

IEEE STD IEEE
Standard

Display Format: ☒ Citation ☐ Citation & Abstract

No results were found.

Please edit your search criteria and try again. Refer assistance revising your search.

Indexed by
 Inspec®

[Home](#) | [Login](#) | [Logout](#)


Welcome United States Patent and Trademark Office

Search Results

[BROWSE SEARCH](#) [IEEE GUID](#)

Results for "~~(((instruction execution instrumenting store restore)<in>n~~
~~<and> (pyr >= 19...~~
 Your search matched 0 documents.
 A maximum of 100 results are displayed, 25 to a page, sorted by Relevance
 Descending order.

» Search Options

[View Session History](#)
[New Search](#)

Modify Search

~~(((instruction execution instrumenting store restore)~~

☐ Check to search only within this results set

» Key

IEEE JNL

IEEE
Journal or
Magazine

IEEE JNL

IEEE Journal
or Magazine

IEEE CNF

IEEE
Conference
Proceeding

IEEE CNF

IEEE
Conference
Proceeding

IEEE STD

IEEE
Standard

Display Format: ☒ Citation ☐ Citation & Abstract

No results were found.

Please edit your search criteria and try again. Refer assistance revising your search.

Indexed by

Inspect

[Home](#) | [Login](#) | [Logout](#)


Welcome United States Patent and Trademark Office

Search Results

[BROWSE SEARCH](#) [IEEE GUID](#)

Results for "~~(((scheduling priority instrumenting store restore)<in>mei~~
~~<and> (pyr >= 1980.~~
 Your search matched **0** documents.
 A maximum of **100** results are displayed, **25** to a page, sorted by **Relevance**
 Descending order.

» Search Options

[View Session History](#)
[New Search](#)

Modify Search

☐ Check to search only within this results set

» Key

IEEE JNL IEEE Journal or Magazine

IEEE JNL IEEE Journal or Magazine

IEEE CNF IEEE Conference Proceeding

IEEE CNF IEEE Conference Proceeding

IEEE STD IEEE Standard

Display Format: ☒ Citation ☐ Citation & Abstract

No results were found.

Please edit your search criteria and try again. Refer assistance revising your search.

Indexed by
 Inspect



instrumentation instruction scheduling store re 1980 - 2

Scholar [All articles](#) [Recent articles](#) Results 1 - 10 of about 1,620 for **instrun**

All Results

[D Scales](#)

[K Gharachorloo](#)

[P Lowney](#)

[J Larus](#)

[D Culler](#)

Shasta: a low overhead, software-only approach for supporting fine-grain shared memory - group of 9 »
DJ Scales, K Gharachorloo, CA Thekkath - Proceedings of the seventh international conference on ..., 1996 - portal.acm.org

... be moved upward from the **instrumentation** point within ... the load when the add **instruction** is executed ... compiler is typically successful in **scheduling** the original ...

[Cited by 262](#) - [Related Articles](#) - [Web Search](#) - [BL Direct](#)

Speculative **register** promotion using advanced load address table (ALAT) - group of 5 »

J Lin, T Chen, WC Hsu, PC Yew - Code Generation and Optimization, 2003. CGO 2003. ..., 2003 - ieexplore.ieee.org

... on the top of ORC to **instrument** the code ... flags are transformed into the corresponding assembly **instructions**. ... code for speculation during list **scheduling** as well ...

[Cited by 15](#) - [Related Articles](#) - [Web Search](#)

Method of reducing the number of overhead instructions by modifying the program to locate ... - group of 3 »

DJ Scales - US Patent 5,758,183, 1998 - Google Patents
... virtual addresses assigned to the memories are allocated to **store** a shared data structure as one or more blocks accessible by **instructions** of programs executing ...

[Cited by 33](#) - [Related Articles](#) - [Web Search](#)

Instruction scheduling for instruction level parallel processors - group of 3 »

P Faraboschi, JA Fisher, C Young - Proceedings of the IEEE, 2001 - ieeexplore.ieee.org

... it is the infamous “job shop **scheduling**” problem [5 ... profiles used to drive **instruction scheduling** as static ... oldest technique is **instrumentation**, where extra ...

Cited by 19 - Related Articles - Web Search - BL Direct

Exploiting hardware performance counters with flow and context sensitive profiling - group of 11 »

G Ammons, T Ball, JR Larus - ACM SIGPLAN Notices, 1997 - portal.acm.org

... have complex microarchitectures that dy- namically **schedule instructions**. ... predictable metrics, such as **instruction** fre- quency ... the effect of **instrumentation** code ...

Cited by 151 - Related Articles - Web Search - BL Direct

The transmeta code morphing software: Using speculation, recovery, and adaptive retranslation to ... - group of 15 »

JC Dehnert, BK Grant, JP Banning, R Johnson, T ... - International Symposium on Code Generation and Optimization, 2003 - doi.ieeecomputersociety.org

... There are **scheduling** constraints that must be ... emulation systems as interpreters (**instruction-at-a-time** ... created for purposes of optimization or **instrumentation**. ...

Cited by 71 - Related Articles - Web Search

Validation checking of shared memory accesses - group of 2 »

DJ Scales - US Patent 5,761,729, 1998 - Google Patents ... of the programs allocates a portion of the virtual shared addresses to **store** a shared data ...

miss handling code is executed before the load
instructions are executed ...

[Cited by 35](#) - [Related Articles](#) - [Web Search](#)

[Register File Design Considerations in Dynamically Scheduled Processors](#) - group of 10 »

KI Farkas, NP Jouppi, P Chow - Proceedings of the Second IEEE Symposium on High-Performance ..., 1996 - doi.ieeecomputersociety.org

... The **instruction scheduling** logic includes a single dis- patch ... **Instruction** counts

are in millions; the “rates ... ing an object code
instrumentation system called ...

[Cited by 75](#) - [Related Articles](#) - [Web Search](#)

[Fine-grain parallelism with minimal hardware support: a compiler-controlled threaded abstract ...](#) - group of 9 »

DE Culler, A Sah, KE Schausser, T von Eicken, J ... - Proceedings of the fourth international conference on ..., 1991 - portal.acm.org

... the initialization thread set up to re- **store** them
from ... the structure to reflect

different **scheduling** policies in ... 21] an I-structure
fetch **instruction** issues a ...

[Cited by 199](#) - [Related Articles](#) - [Web Search](#) - [Library Search](#)

[Open research compiler \(orc\) 2.0 and tuning performance on itanium](#)

R Ju, S Chan, TF Ngai, C Wu, Y Lu, J Zhang - 35th International Symposium on Microarchitecture, December, 2002 - cs.ualberta.ca

... gap from source language to machine **instructions** •

Same IR ... eg MISTORE – indirect

store of memory chunk • Field ID ... **Register Variable Identification II** ...

[Cited by 4](#) - [Related Articles](#) - [View as HTML](#) - [Web Search](#)

Goooooooooooooogle ►

Result Page: 1 2 3 4 5 6 7 8 9 10 [Next](#)

instrumentation instruction schedulir

Search

[Google Home](#) - [About Google](#) - [About Google Scholar](#)

©2007 Google